Data Science Homework 1

Group 32: Michael Wang, Praveen Gupta, Kerem Tutuncu

May 5, 2018

1 ETL, Feature Extraction, Data Preparation

1.1 Summary Statistics

We computed the following statistics for numerical features on the entire sample dataset of 100k samples:

Feature	Mean	Standard Deviation
1	3.76872290284	10.4511149082
2	112.86373	401.520628263
3	40.7449133477	538.815492135
4	8.28031715259	10.836268528
5	17592.5994015	65797.5526356
6	139.685084053	371.773609476
7	15.2220904483	65.4601431021
8	13.5748250628	46.5413599367
9	125.294906645	286.414183229
10	0.620109018296	0.677049264484
11	2.40026867896	4.62990171541
12	0.937773882559	5.32755161989
13	11.6076355231	52.0442325898

Please refer to the Appendix for histogram plots of all the features.

1.2 Data Partitioning

To partition the data we took advantage of spark's `random_split` function which randomly samples partitions of the dataset. We used the sample set of 100k and split it into partitions of 50% training, 20% validation, and 30%. We used the sample set for all of our calculations since we ran into memory errors that we could not resolve on our local machines for larger sets (See section 3).

1.3 Normalization and Feature Engineering

To normalize numerical features, we computed the mean and standard deviation of the training set and standardized the data by subtracting the mean and dividing by the standard deviation. We applied the same transformation to the test set.

Since many of the samples were missing values in certain features, we decided to drop features for which too many samples were missing values. Specifically, we thresholded such that if 30% of samples were missing a feature, we dropped the feature. Figure 1 shows a histogram of the number of samples missing values for each feature. This was a trade off between keeping as many of the samples as possible without skewing the data since we would have to replace any remaining missing values. For the remaining null values, we substituted in the training set mean for numerical features, and a special 'nullvalue' category for categorical features.

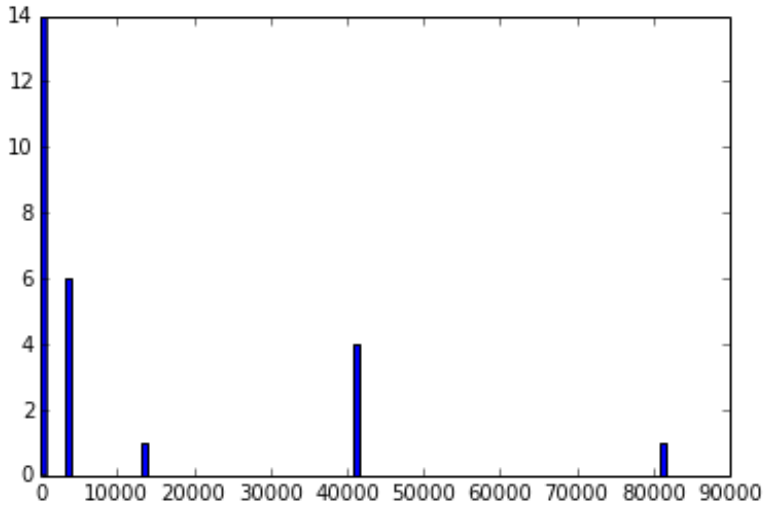


Figure 1: Missing value histogram for features

Another method for feature selection we considered was dropping features that had very biased distributions. By examining the histograms of frequencies for values and categories, we would drop features where a small percentage of values comprised the majority of the density. Finally, given more time we would have explored Chi-Squared feature selection, which uses the Chi-Squared test to eliminate features that are likely to be independent from the classes labels.

We encoded categorical variables with one-hot-encoding.

2 Modeling and Evaluation

To classify the test set, we ran two classification models: Random forest and logistic regression.

2.1 Random Forest Results

3 trees, max depth = 2:
Accuracy: 0.540500440209

30 trees, max depth = 20:
Accuracy: 0.724223564784

200 trees, max depth = 50:

Accuracy: 0.740979703797
AUC: 0.5

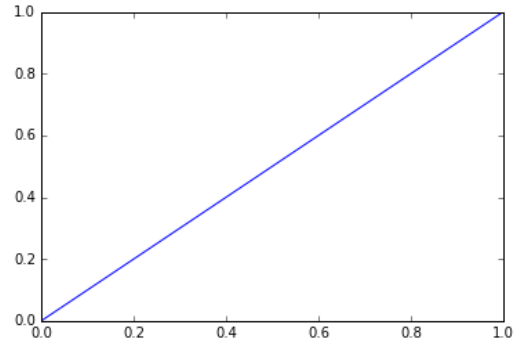


Figure 2: Random Forest ROC Curve

2.2 Logistic Regression Results

Iterations = 10, regParam=0.3, elasticNetParam=0.8:
Accuracy: 0.5

Iterations = 100, regParam=0.01, elasticNetParam=0.8:
Accuracy: 0.677504714772

Iterations = 100, regParam=0.01, elasticNetParam=0.1:
Accuracy: 0.709508756355
AUC: 0.510460171522

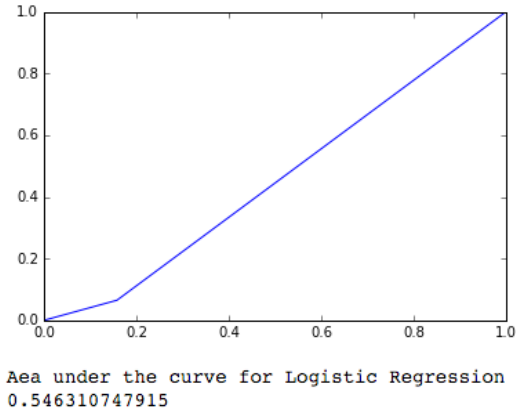


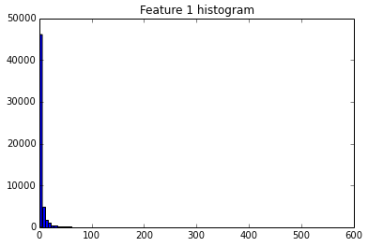
Figure 3: Logistic Regression ROC Curve

3 Pain points

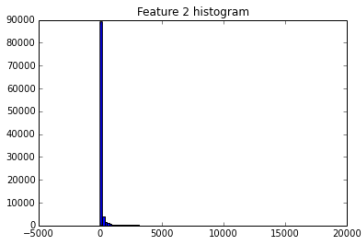
We ran into a variety of problems using Spark's built in functions and datatypes that ultimately took up our time and resources during this assignment. We started off using the newer Spark ml library as opposed to the mllib library. This API uses Dataframes for the most part, which proved much more difficult to work with. The reason we went with the ml API was to take advantage of many of the powerful functions to streamline the flow, such as Pipeline to chain multiple transformations. The API also had OnehotEncoder, StringIndexer, and VectorAssembler which should have make the datastructures easier to work with. However, as we discovered, the library is not well maintained and we ran into errors that were unresolvable. For example, the ml version of StandardScaler runs into issues with SparseVectors and data type conversion was a nightmare with Dataframes. Speaking with other groups, it seems many just used RDDs and implemented some of these functions manually which actually achieved faster speeds than the library ones working with dataframes. Additionally, we ran into memory issues, even when running just the 100k sample set. We allocated more memory by configuring the spark driver and executor memory settings, made sure not to call collect anywhere in our code, and also removed all persist() calls, but these did not fix the issue. In the future we will look into sticking with RDDs.

4 Appendix

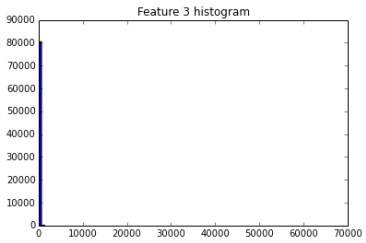
4.1 Feature Histograms



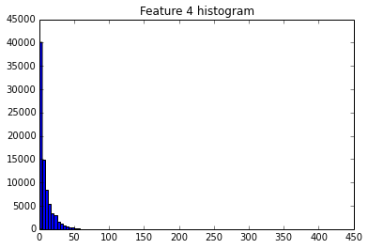
(a) Feature 1 histogram



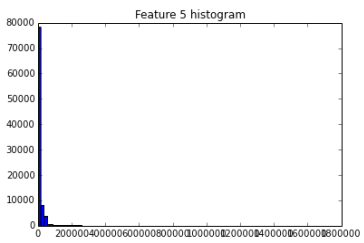
(b) Feature 2 histogram



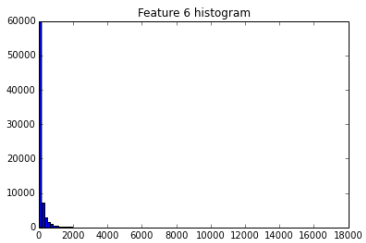
(c) Feature 3 histogram



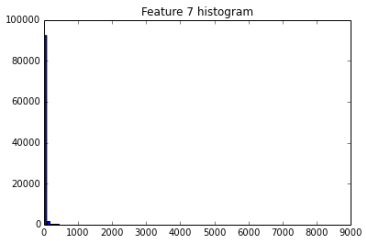
(a) Feature 4 histogram



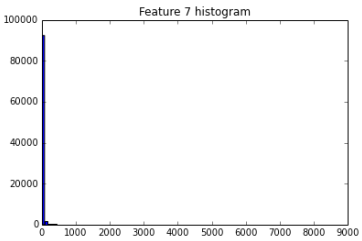
(b) Feature 5 histogram



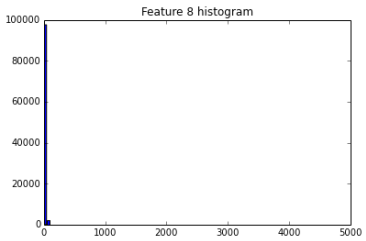
(c) Feature 6 histogram



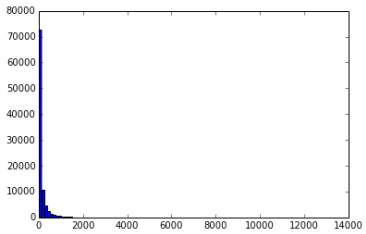
(a) Feature 7 histogram



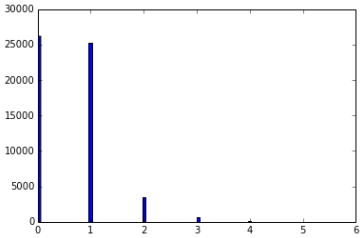
(b) Feature 7 histogram



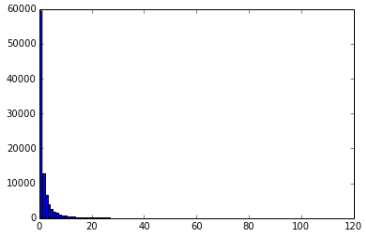
(c) Feature 8 histogram



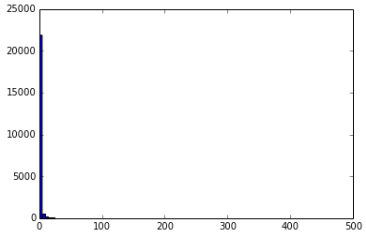
(a) Feature 9 histogram



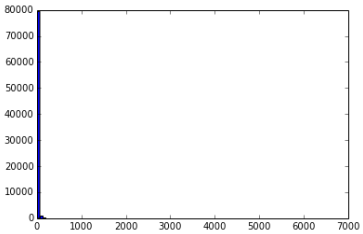
(b) Feature 10 histogram



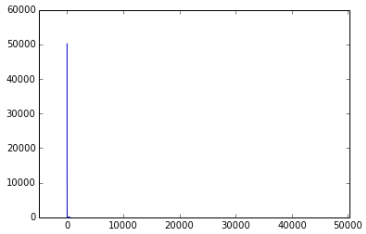
(c) Feature 11 histogram



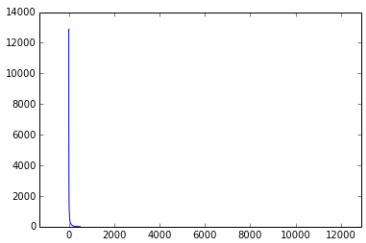
(a) Feature 12 histogram



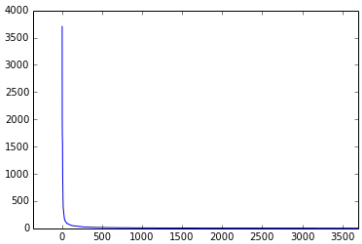
(b) Feature 13 histogram



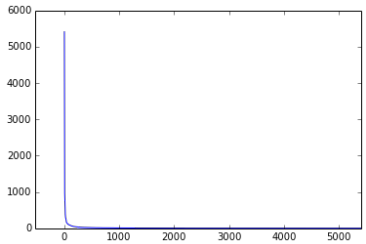
(c) Feature 14 histogram



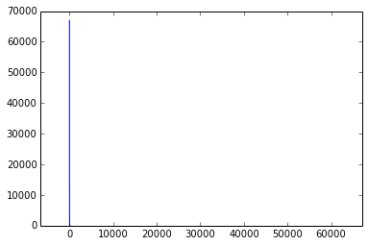
(a) Feature 15 histogram



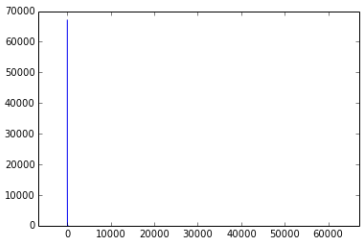
(b) Feature 16 histogram



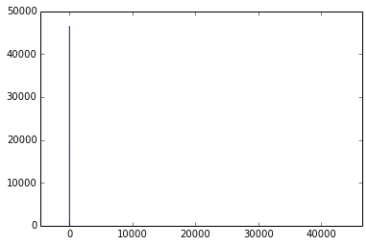
(c) Feature 17 histogram



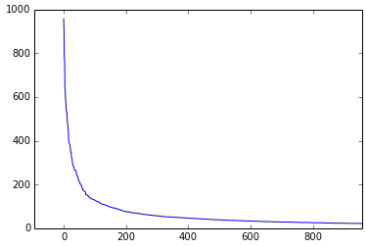
(a) Feature 18 histogram



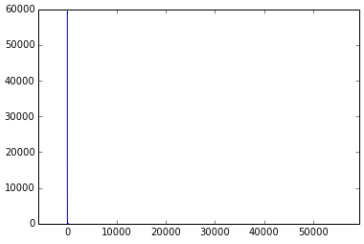
(b) Feature 18 histogram



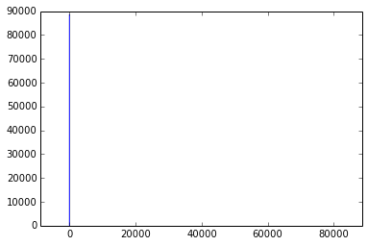
(c) Feature 19 histogram



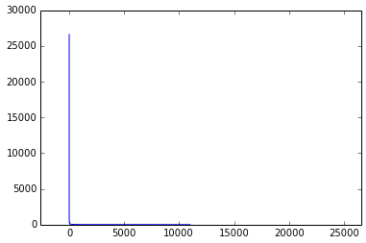
(a) Feature 20 histogram



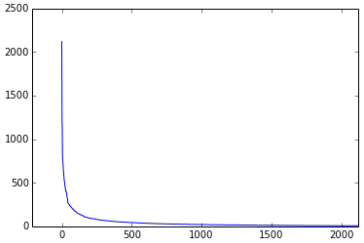
(b) Feature 21 histogram



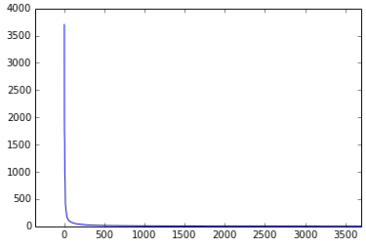
(c) Feature 22 histogram



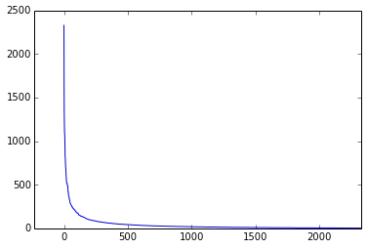
(a) Feature 23 histogram



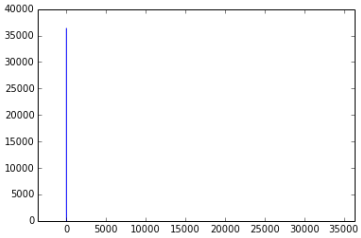
(b) Feature 24 histogram



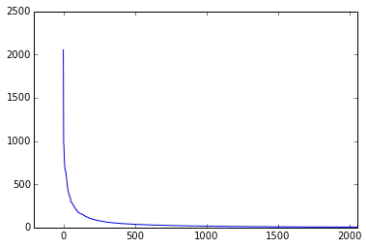
(c) Feature 25 histogram



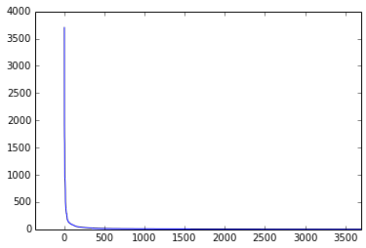
(a) Feature 26 histogram



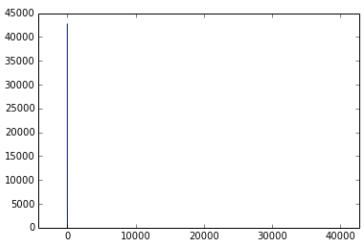
(b) Feature 27 histogram



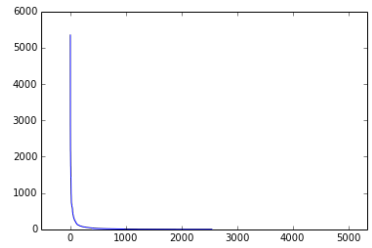
(c) Feature 28 histogram



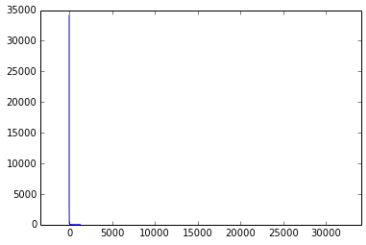
(a) Feature 29 histogram



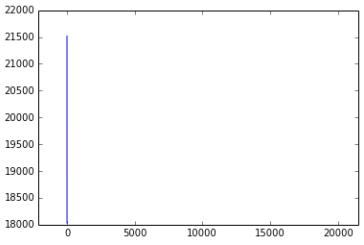
(b) Feature 30 histogram



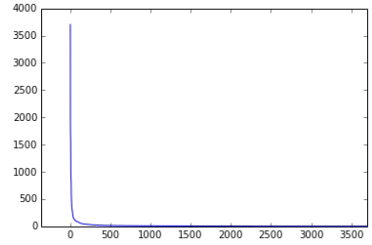
(c) Feature 31 histogram



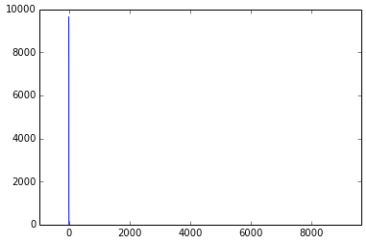
(a) Feature 32 histogram



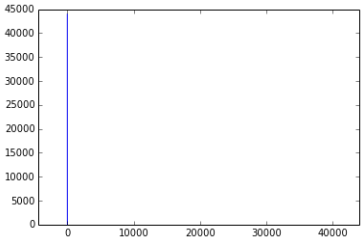
(b) Feature 33 histogram



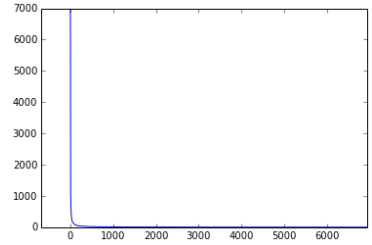
(c) Feature 34 histogram



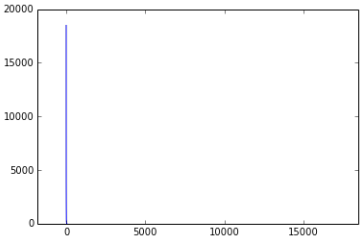
(a) Feature 35 histogram



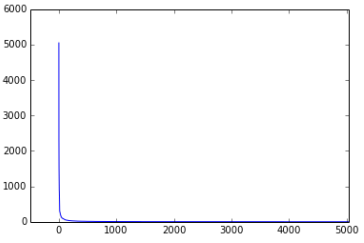
(b) Feature 36 histogram



(c) Feature 37 histogram



(a) Feature 38 histogram



(b) Feature 39 histogram